Sequential Routing Framework: Fully Capsule Network-based Speech Recognition

Kyungmin Lee[a,b], Hyunwhan Joe[a], Hyeontaek Lim[b], Kwangyoun Kim[b], Sungsoo Kim[b], Chang Woo Han[b], Hong-Gee Kim[a,*]

[a] *Biomedical Knowledge Engineering Laboratory, Seoul National University, Seoul, 08826, Republic of Korea*
[b] *Samsung Research, 56, Seongchon-gil, Seocho-gu, Seoul, 06765, Republic of Korea*

Abstract

Capsule networks (CapsNets) have recently gotten attention as alternatives for convolutional neural networks (CNNs) with their greater hierarchical representation capabilities. In this paper, we introduce the sequential routing framework (SRF) which we believe is the first method to adapt a CapsNet-only structure to sequence-to-sequence recognition. In SRF, input sequences are capsulized then sliced by the window size. Each sliced window is classified to a label at the corresponding time through iterative routing mechanisms. Afterwards, training losses are computed using connectionist temporal classification (CTC). During routing, two kinds of information, learnable weights and iteration outputs are shared across the slices. By sharing the information, the required parameter numbers can be controlled by the given window size regardless of the length of sequences. Moreover, the method can minimize decoding speed degradation caused by the routing iterations since it can operate in a non-iterative manner at inference time without dropping accuracy. We empirically proved the validity of our method by performing phoneme sequence recognition tasks on the TIMIT corpus. The proposed method attains an 82.6% phoneme recognition rate. It is 0.8% more accurate than that of CNN-based CTC networks and on par with that of recurrent neural network transducers (RNN-Ts). Even more, the method

*Corresponding author
Email address: hgkim@snu.ac.kr (Hong-Gee Kim)

requires less than half the parameters compared to the two architectures.

Keywords: Capsule network, Automatic speech recognition,
Sequence-to-sequence, Connectionist temporal classification

1. Introduction

Capsule networks (CapsNets) (Hinton et al., 2011; Sabour et al., 2017; Hinton et al., 2018) are a kind of neural network that represents a specific entity type by a group of neurons, which are referred to as capsules, instead of a single neuron represented as a scalar. The initial motivation of CapsNets was to abstract information more explicitly by adapting an unsupervised clustering method called routing-by-agreement to neural networks. By introducing the new representation units, the routing mechanism can be performed between adjacent capsules during neural network training. Capsules can represent not only the existence of the entity types but also entity instantiation parameters such as textures, angles, colors, and so on, thus CapsNets can be regarded as inverse rendering architectures. They have received notable attention as promising alternatives to convolutional neural networks (CNNs). Particularly, this is because the pooling layers in CNNs filter information within each patch from lower to higher layers based on fixed operations such as maximum or average, unlike the adaptive mechanism i.e. routing mechanisms in CapsNets. CapsNets have shown higher accuracy in image classification on MNIST (LeCun and Cortes, 2010) and Canadian Institute For Advanced Research (CiFAR) 10 (Krizhevsky, 2009) than CNNs (Hahn et al., 2019; Malmgren, 2019). Previously, research on CapsNets have mostly focused on improving the performance of routing methods (Malmgren, 2019; Wang and Liu, 2018) on smaller datasets. Recently, research has been focused more on modifying the routing methods to fit more practical problems such as scaling the routing methods itself (Tsai et al., 2020) or by combining them with other architectures (He et al., 2019a; Srivastava et al., 2019; Wang, 2019). There have also been attempts to apply CapsNet to various fields beyond image classification such as user intent detec-

tion (Xia et al., 2018; Zhang et al., 2019a), self-driving (Kim and Chi, 2019), sound event detection (Iqbal et al., 2018), speech command classification (Bae and Kim, 2018) and speech emotion recognition (Wu et al., 2019).

Sequence to sequence (seq2seq) learning is an approach to learn mappings between input and output sequences and learning is successfully implemented by neural models (Sutskever et al., 2014). It is one of the most popular methods in a wide range of fields such as automatic speech recognition (ASR) (Chan et al., 2016; He et al., 2019b; Kim et al., 2019; Moritz et al., 2019), machine translation (MT) (Bahdanau et al., 2015; Jia et al., 2019), natural language processing (NLP) (Yin et al., 2016), etc. Connectionist temporal classification (CTC) (Graves et al., 2006) is a popular loss function in seq2seq problems since the method can compute a conditional probability between a single reference and multiple possible paths of a network output vector sequence. As CTC started to be applied to ASR tasks, it became possible to learn the alignment between speech signal sequences and label sequences directly unlike the conventional Hidden Markov Model (HMM)-based methods (Rabiner, 1989) that need separately aligned intermediate representations. Recurrent neural network transducers (RNN-Ts) (Graves et al., 2013) have expanded CTC by introducing an additional axis to represent linguistic information. They are now one of the most pervaded methods in commercial ASR systems (He et al., 2019b). Recently, CTC networks are also used as a form of joint training (Kim et al., 2019; Moritz et al., 2019) with cross-entropy based networks to help encoders learn better alignments when training attention-based encoder-decoder models.

In seq2seq problems, long short term memory (LSTM) (Hochreiter and Schmidhuber, 1997) is one of the most common approaches. LSTMs are a kind of recurrent neural networks (RNNs) and have better sequence encoding capabilities than vanilla RNNs since they can learn more explicitly how to filter contextual information with cell states and gating mechanisms implemented as non-linear activation functions. CTC networks were first built based on LSTMs (Graves et al., 2006) and they have shown competitive performance compared to traditional ASR methods. In particular, bidirectional LSTM (BLSTM)-based

CTC networks have shown the most accurate recognition rate among the early implementations (Graves et al., 2013). In order to relieve computational burden and avoid the vanishing and exploding gradient problem of LSTMs, a CNNs-based CTC network (Zhang et al., 2016) has been proposed as an alternative. CNNs encode the input feature sequences by filtering the context information with their own mechanisms which convolve the sequences with multiple filters to detect specific shapes. They then down-sample the convolved sequences with pooling methods as explained earlier in this section. Through this hierarchical structure, they can encode spatial relations wider than each filter size with deeper convolutional layers. Recently, Transformers which are self-attention-based neural architectures (Vaswani et al., 2017), have been in the spotlight as another alternative to the recurrent models. Transformers encode sequences by filtering with attention maps which indicate within-sequence similarities calculated with scaled dot-products. Although Transformers do not have an appropriate structure for streaming decoding since they need the whole context to compute the attention map, there have been recent attempts to apply them to streaming decoding (Moritz et al., 2020). With their own context filtering mechanisms, the two alternative methods have shown similar or better accuracy compared to LSTMs in ASR related fields (Zhang et al., 2016; Dong et al., 2018; Zeghidour et al., 2018; Moritz et al., 2020) while requiring less computations and have suitable structures for parallelization. In particular, the CNN-based CTC networks (Zhang et al., 2016) have shown 0.2% more accurate phoneme recognition rate (PRR) with 63.2% of parameters compared to the BLSTM-based architecture while maintaining online decoding capabilities. The input feature sequences can be regarded as two dimensional images, having the time length as width and the feature dimension as height. In this perspective, we hypothesized that CapsNets can potentially have richer representation capabilities than CNNs also in seq2seq problems for the same reasons mentioned earlier in this section. Research on CapsNets have mostly focused on classification tasks with fixed input sizes as alternatives for CNNs. Although there have been attempts to apply CapsNets to seq2seq tasks in combination with other mod-

els (Srivastava et al., 2019; Wang, 2019), the utilization of CapsNets could be maximized more. There are several hurdles in applying existing CapsNet-only architectures to seq2seq problems without modification. This is because routing from the variable-length capsule inputs to the sequential combinations of labels is infeasible in the perspectives of both computation and the required parameter amount. Furthermore, ASR systems which are one of the most common seq2seq applications essentially require real-time processing, which can be hampered by the computationally expensive iterative routing mechanisms.

In this paper, we introduce the sequential routing framework (SRF) which is a novel method to build up CTC networks based on a CapsNet-only architecture. SRF is applicable to iterative routing-by-agreement methods (Sabour et al., 2017; Hinton et al., 2018) which update the routing coefficients of the current iteration using the outputs of the previous iteration in an expectation and maximization manner. In order to validate our proposed method, we performed phoneme sequence recognition by applying Dynamic routing (DR) (Sabour et al., 2017) to the framework because it has shown competitive performance (Hahn et al., 2019; Malmgren, 2019). To train SRF models, the input feature sequences for the framework are first converted to three-dimensional sequences through convolutional and linear projection layers. The encoded sequences are sliced by time windows and multiple routing iterations are performed for each slice to classify the corresponding label. Afterwards, the training loss is calculated using CTC. The framework achieved competitive accuracy while minimizing decoding speed degradation and required parameters by temporally sharing two types of information during routing. In this paper, what we mean by "temporally sharing" is to share information across the time slices. First, the learnable variables for computing the path representations from lower to higher levels are shared so that only the fixed size of parameters, according to the given window size, is required. In addition to the weight sharing, the capsule clustering information is also conveyed to the next time slice by initializing routing coefficients of the current slice based on the previous routing results. As a result, iteration is not required during inference meanwhile the

routing coefficients can have an effect similar to be updated by the number of times corresponding to each window time slice.

A clear contribution of this study is that, to the best of our knowledge, SRF is the first CapsNets only architecture for the seq2seq recognition problem. The proposed method achieved competitive performance on the speech recognition task in three aspects that are accuracy, decoding speed, and parameter numbers by temporally sharing learnable weights and clustering information.

2. Preliminaries

2.1. CTC Network

A CTC network (Graves et al., 2006) is defined as a continuous map $\mathcal{N}_\theta : (\mathbb{R}^F)^T \mapsto (\mathbb{R}^V)^T$ from an F dimensional input sequence x of length T to the same length sequence $\hat{y}$ of V dimensional probability vectors with parameters θ. V is the cardinality of an expanded label set $\mathbb{L}'$ consisting of the union between the label symbol set $\mathbb{L}$ and the blank symbol set, $\mathbb{L} \cup \{<\text{blank}>\}$. This network contains a softmax layer at the top of it in order to convert logits to valid probability distributions. The softmax layer for a given vector $Z \in \mathbb{R}^D$ is computed as follows:

$$\text{softmax}(Z)_i = \frac{\exp(z_i)}{\sum_{j=1}^{D} \exp(z_j)}, \text{for } i = 1, ..., D \tag{1}$$

, where z_i is the i-th element of Z. CTC computes a conditional probability of a label sequence $y \in \mathbb{L}^{\leq T}$ for a given input sequence x by summing up every conditional probability of possible paths π, i.e. $p(y|x) = \sum_{\pi \in \mathcal{B}^{-1}(y)} p(\pi|x)$. The possible paths are computed using an inverse of a map $\mathcal{B} : \mathbb{L}'^T \mapsto \mathbb{L}^{\leq T}$ from an expanded label sequence y' to y. $\mathcal{B}$ performs many-to-one mappings by simply removing repeating and blank symbols from the given paths. For example, $\mathcal{B}$(“cc-aaa-tt”) = $\mathcal{B}$(“-cc-aattt”) = “cat”, where “-” indicates a blank symbol. It allows CTC networks to learn alignments solely from the input and output sequence pairs. A conditional probability of each possible path is computed as $p(\pi|x) = \prod_{t=1}^{T} p(y'^t_\pi|x^t)$, where y'^t_π is an expanded label symbol in a path π at

time t and x^t is the t-th feature vector of x. An objective function is defined to compute a negative summation of log probabilities of all the possible paths corresponding to the given labeling and CTC networks are trained to minimize it. The gradient of the objective function can be computed with respect to $\hat{y}$ with any kind of gradient-based optimization techniques. To derive the gradient more efficiently, the CTC forward-backward algorithm, which is a kind of dynamic programming, is applied during training.

2.2. Capsule Network

The main structural difference of CapsNets with other neural architectures is that they use a group of neurons as an element of representation rather than a single neuron (Hinton et al., 2011). The group of neurons is referred to as a capsule and it consists of an activation scalar and a group of instantiation parameters. For a certain entity type, the former indicates an existence probability while the latter represents characteristics such as deformations, poses, and textures in the form of a vector or matrix. Thus, encoding image data through CapsNets can be seen as a mechanism of inverse renderings which converts input data to their instantiation parameters. The activations can be computed either from the instantiation parameters (Sabour et al., 2017) or from additional neural layers (Hinton et al., 2018), according to the routing mechanisms. With the two representations, CapsNets can potentially (Lenssen et al., 2018) represent invariance of the existence probabilities and also equivariance of the properties of the entity type. The input and output of capsule layers are the group of capsules which have a three-dimensional structure. Accordingly, a capsule group is composed of a pair of an instantiation parameter vector group and its corresponding activation scalar group. In this paper, for the sake of brevity, we describe the structure of an instantiation vector group in a capsule group as the structure of the capsule group because the structure of the activation scalar group can be derived by just removing the depth of the vector group structure. The product of the width and height of a capsule group indicates the number of capsules in the group and their depth represents the dimension of the instan-

tiation parameter vectors. The capsule groups in the lowest, the highest, and in-between levels of CapsNets are called as primary, class, and convolutional capsule groups respectively. The width of class capsule groups is one and its height is equal to the number of labels for classification tasks. In order to compare the class capsule group with the given label, each instantiation vector of the group is flatten to an activation i.e. the final output structure of a CapsNet is converted to the same as that of a conventional neural network. Thus any loss methods used for other neural networks are applicable for CapsNets without modification.

One of the key abilities of neural networks is to abstract the information in input data to higher concepts layer by layer, in order to make them a suitable representation for the purpose of training. We can assume that this hierarchical conceptualization can be seen as a tree-like structure made of multiple part-whole relationships. In order to use this assumption more explicitly to train models, CapsNets introduce the routing-by-agreement method which is a kind of unsupervised clustering algorithm. The method routes information from lower to higher levels based on the agreements of capsules. In this perspective, the routing mechanism can be thought of as an adaptive layer-wise filtering method compared to both static filtering such as striding or pooling of CNNs and sequence-wise filtering such as the gates of LSTMs or self-attentions of Transformers. The one key procedural difference is that routing-by-agreement is a non-gradient based learning algorithm. Research related to CapsNets have mainly focused on iterative routing methods which update their clustering mechanisms based on the outputs of previous iterations (Sabour et al., 2017; Hinton et al., 2018). Recently in order to reduce their computational burdens, non-iterative (Hahn et al., 2019) or parallelized (Tsai et al., 2020) methods have also been introduced.

2.2.1. Dynamic Routing

DR (Sabour et al., 2017) is a kind of iterative routing-by-agreement method that controls information flow from lower to higher levels based on the capsule

similarities between the adjacent capsule groups in a non-parametric manner. It is one of the most widely used implementation of CapsNets because of its competitive accuracy and intuitive mechanisms compared to more recent architectures (Hahn et al., 2019; Malmgren, 2019). In DR, activation scalars of capsules are defined by the lengths of the instantiation parameter vectors so that they need to be normalized to have bounded lengths within the range from zero to one. In order to normalize them, a nonlinear function, which is referred to as a squash function, is defined as follows:

$$\text{squash}(s_j) = \frac{||s_j||^2}{1 + ||s_j||^2} \frac{s_j}{||s_j||} \tag{2}$$

, where s_j is a unnormalized instantiation parameter vector for the j-th capsule of the output capsule group. This function also has a role to make the output activations more discriminative by pushing most of the values to around zero or one. s_j is computed as $s_j = \sum_i^{l_N} c_{ij} \hat{u}_{j|i}$, that is the summation of the prediction vector $\hat{u}_{j|i}$ from the i-th capsule in the lower level to the j-th capsule in the higher level weighted with corresponding coupling coefficients c. The coefficients are normalized values through the softmax function described in Eq. 1 from the routing coefficients r, which is updated by a routing method. $\hat{u}_{j|i}$ is calculated as the product of an instantiation parameter vector u_i, which is the i-th capsule in an input capsule group, and a learnable weight matrix W_{ij}. Thus the required number of W for each routing mechanism is the product of the capsule numbers of input and output capsule groups and each W_{ij} has the shape of the input capsule depth times the output capsule depth. The primary capsules are calculated from input data through multiple convolutional layers activated with rectified linear unit (ReLU) functions.

In the routing algorithm, r is initialized uniformly, then it is iteratively updated to maximize the agreements between $\hat{u}_{j|i}$ and the output vector o_j by accumulating the values of their products. In order to allow for multiple digits, a separate margin loss between labels and the activations of the final output vectors is calculated as a prediction error. Reconstruction errors are calculated as the mean squared errors between the original signals and the reconstructed

signals generated from the output capsule corresponding to the target label through additional feed forward layers. A total loss for training is computed by adding a reconstruction error to the prediction error as a regularization term.

3. Sequential Routing Framework

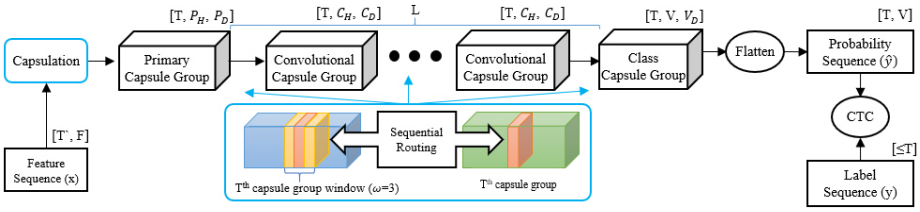


Figure 1: An overview of the sequential routing framework (SRF)

SRF is an iterative routing framework for sequence data and it is defined as a modified version of CTC networks with parameter ψ, $\mathcal{S}_\psi : (\mathbb{R}^{F'})^{T'} \mapsto (\mathbb{R}^V)^T$, from a real valued feature sequence x having a length T' and a feature dimension F' to a V dimensional probability vector sequence $\hat{y}$ of a width T as shown in Fig. 1. The shapes of an input or output sequence and each capsule group are written above the boxes in the diagram. A two-dimensional input feature sequence x is transformed into a primary capsule group, whose width, height and depth are T, P_H and P_D respectively, through a capsulation block. Afterwards, the primary capsule group is fed into the lowest capsule layer, then encoded to a convolutional capsule group, whose width, height and depth are T, C_H, and C_D respectively. For the sake of brevity, we describe all the convolutional capsule groups have the same shape. The output of the L-th capsule layers, i.e. a class capsule group, has T, V, and V_D as width, height and depth respectively. The capsule group is flattened to an activation vector sequence $\hat{y}$ which is a sequence of probability vectors where each element indicates a probability of observing a corresponding label symbol. If L is set to one, the lowest capsule layer directly outputs class capsule groups. Finally, a negative log probability is computed using CTC between $\hat{y}$ and a given label sequence y as a loss for training the neural network.

3.1. Capsulation

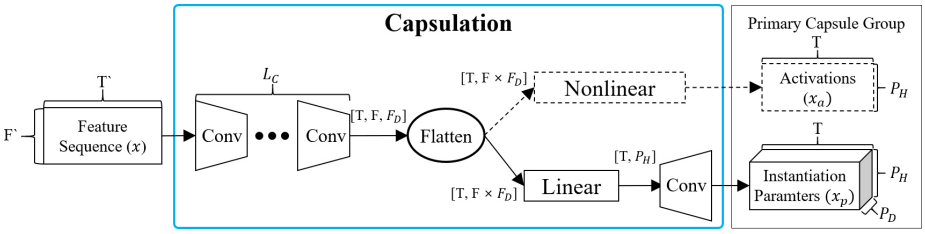


Figure 2: A capsulation block

Capsulation is a neural layer block that converts from x to a group of primary capsules which consists of two subgroups as shown in the Fig. 2. One group is an activation group with dimensions of width T and height P_H and the other is an instantiation parameter group with width T, height P_H and dimension P_D. The layers to compute the group of activations are optional structures depending on the routing mechanism. A feature sequence x is encoded into three-dimensional representation with width T, height F and depth F_D through L_C two-dimensional convolutional layers, where $T \leq T'$ and $F \leq F'$. We employed maxout (Goodfellow et al., 2013) as activation functions of the convolutional layers because of their competitive accuracy in ASR systems (Zhang et al., 2016). Thus, the l_c-th convolutional layer in the first sub-block is defined as follows:

$$x_{l_c} = \max_{n_c \in [1, N_C]} \text{Dropout}_{n_c}([W_{l_c, n_c, f_d} * x_{l_c - 1}]_{f_d=1}^{F_D}, \alpha_d),\ l_c \in [1, L_C] \qquad (3)$$

, where $*$ is a convolutional operator, N_C is the number of convolutional operations in a layer and α_d is the dropout rate. W_{l_c, n_c, f_d} indicates a convolution weight matrix for the f_d-th channel of the n_c-th convolutional operation in the l_c-th layer. The 0-th x is an input sequence, i.e. $x_0 \in (\mathbb{R}^{F'})^{T'}$. For all the cases, we set N_C and α_d to 2 and 0.2 respectively. Subsequently, the output sequence is flattened to x'_{L_C} by simply reshaping it to width T and height $F \times F_d$, then it is fed into two kinds of layer blocks. One of them is projected to a normalized vector sequence, i.e. an activation group x_a, with width T and height P_H,

through a nonlinear layer as follows:

$$x_a = \sigma(W_a \times x'_{L_C}) \tag{4}$$

, where W_a is a learnable weight matrix and σ is a nonlinear function for normalizing the values. The other is projected to another representation having the same shape with the activation group as follows:

$$x'_p = W_p \times x'_{L_C} \tag{5}$$

, where W_p is a learnable weight matrix. Afterwards, it is converted to a group of instantiation parameters x_p by expanding their channel dimensions into P_D through a two-dimensional convolutional layer activated with the maxout function as follows:

$$x_p = \max_{n_c \in [1, N_C]} \text{Dropout}_{n_c}([W_{n_c, p_d} * x'_p]_{p_d=1}^{P_D}, \alpha_d) \tag{6}$$

, where W_{p_d} indicate a convolutional weight matrix for the p_d-th output channel of the n_c-th convolutional operation. We set N_C and α_d identically for every layer activated with maxout in order to reduce the number of hyperparameters.

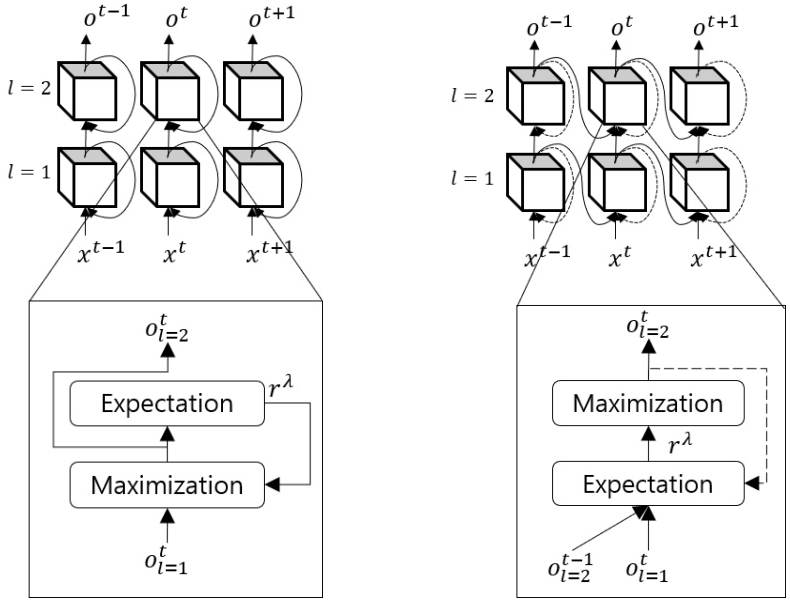
3.2. Routing-by-agreement for sequence to sequence learning

In SRF, each subgroup of a capsule group $\{x_a, x_p\}$ is sliced by t, i.e. $x^t = \{x_a^t, x_p^t\}$. The routing mechanism which we will refer to as sequential routing is performed for each slice. In order to expand the size of a receptive field on a primary capsule group, sequential routing is performed between window slices, which consist of ω slices of the lower level, and single slices of the higher level. The bottom box in Fig. 1 describes an example of sequential routing between the t-th window slice ($\omega = 3$) in the lower level and the t-th single slice in the higher level. In this study, we set the stride of the sliding window to one in all the cases and the both side of window contexts beyond the sequence boundaries are padded as zero so that the routing is performed T times for a primary capsule group having width T. The width of the receptive field on a primary capsule group is computed as $\omega + (L-1) \times (\omega - 1)$ for each slice of the output of the

L-th capsule layer. Accordingly, online decoding is allowed with a time delay corresponding to $L \times \omega_R$, where ω_R is the right context size of the window. In each capsule layer, the prediction vector $\hat{u}_{j|i}$ is calculated per each t through a linear transformation consisting of a learnable weight matrix W_{ij} as follows:

$$\hat{u}^t_{j|i} = W_{ij} \times x^t_{p_i}, i \in [1, \omega \times I_H], j \in [1, O_H] \tag{7}$$

, where I_H and O_H are heights of input and output capsule groups respectively and each W_{ij} has the shape of the product of dimensions of input and output capsule groups. Consequently, the number of parameters to perform the routing method in each capsule layer is controlled by the given three kinds of hyper parameters, that are the height and depth of capsule groups and ω, regardless of the input sequence length since the parameters are temporally shared across all t.



(a) Original routing (b) Sequential routing

Figure 3: Schematic diagrams of the two versions of iterative routing procedures ($L = 2$, $\omega = 1$, $\lambda \in [1, \Lambda]$).

We have an assumption that consecutive slices in sequence data have similar

properties. Based on that, we designed an iterative routing mechanism which initializes routing coefficients of the t-th iteration based on the $(t-1)$-th iteration outputs. Fig. 3 is a schematic diagram of the two different iterative routing mechanisms of $L = 2$ and $\omega = 1$ for three consecutive window slices. In this figure, the three dimensional block represents the routing procedures, λ is an iteration index in the range between 1 and the given iteration number Λ, and the solid and dashed arrows describe the required and optional flows to update routing coefficients r respectively. In the original version of routing, r is initialized uniformly, then updated within each t iteratively as in Fig. 3a in the order of a maximization and an expectation stage. The expectation stage means to update r to improve the agreements between adjacent capsule groups and the updated coefficients are applied in the maximization stage. The proposed sequential routing method has two procedural differences from the original version as described in Fig. 3b. First, r is initialized based on the agreements between the previous routing outputs o_l^{t-1} and the current routing inputs o_{l-1}^t thus r is uniformly initialized only at $t = 1$. The second procedural difference is the sub-procedure order of the routing mechanism which expects the initial r before the first maximization stage. These two modifications have a similar effect of updating r for $(t-1)$ times to compute the t-th output o^t when Λ for each window is set to one. In other words, they alleviate the need for iterative updates of r within each slice as an option. Consequently, decoding can be performed in a non-iterative way when Λ is set to one.

In this research, we adapt DR to SRF and it works as Algorithm 1 for each t between the l-th and $(l+1)$-th level. The previous output vector o_j^{t-1}, $\hat{u}_{j|i}$, Λ, and the lower level index l are given as inputs to the algorithm as in the first line of the algorithm. r is set to zero at line 2 and 3. The current output vector o^t is initialized to o^{t-1} at line 4. Afterwards, the expectation-maximization clustering algorithm is performed from line 5 to 11. At lines 6 and 7, r is updated by accumulating the agreements of $\hat{u}_{j|i}^t$ and o_j^t, i.e. their products. The updated coefficients are normalized to c using the softmax function, which is Eq. 1, at line 8. At line 9, in order to maximize the agreements using the

updated coefficients, the unnormalized output s_j is computed as the summation of $\hat{u}^t_{j|i}$ over all i by weighting with c_{ij}. The length of s_j are normalized using the squash function Eq. 2. $o^{t=0}$ is set to a zero vector in order to initialize c uniformly for $t = 1$. The number of operations of an iteration remains the same as the original version of DR since only the order of the expectation and maximization sub-procedures is changed. At the top layer, coefficients in c which route capsules to a class capsule corresponding to the padding symbol are masked as zero.

Algorithm 1 Sequential version of Dynamic Routing (DR) algorithm

procedure SEQUENTIAL DYNAMIC ROUTING($o^{t-1}_j, \hat{u}^t_{j|i}, \Lambda, l$)
  for all capsule i in t-th window of level l
    and capsule j in t-th slice of level $(l+1)$: $r_{ij} \leftarrow 0$
  for all capsule j in t-th slice of level $(l+1)$: $o^t_j \leftarrow o^{t-1}_j$
  for Λ iterations **do**
    for all capsule i in t-th window of level l
      and capsule j in t-th slice of level $(l+1)$: $r_{ij} \leftarrow r_{ij} + \hat{u}^t_{j|i} \cdot o^t_j$
    for all capsule i in t-th window of level l: $c_i \leftarrow \text{softmax}(r_i)$ ▷ eq 1
    for all capsule j in t-th slice of level $(l+1)$: $s_j \leftarrow \sum_i c_{ij} \hat{u}^t_{j|i}$
    for all capsule j in t-th slice of level $(l+1)$: $o^t_j \leftarrow \text{squash}(s_j)$ ▷ eq 2
  end for
end procedure

4. Results

4.1. Experiment Setup

4.1.1. Data preparation

TIMIT (Garofolo et al., 1993) consists of mono-channel read speech sampled at 16Khz. The training and test set consist of 4,620 utterances recorded from 462 speakers and 1,680 utterances recorded from 168 speakers respectively. We used a training set consisting of 3,696 utterances where all dialect utterances,

i.e. the utterances tagged as "SA", were removed and used 192 core test sets recorded from 24 speakers. A validation set was selected from another portion of the test set and was made up of 400 utterances recorded from 50 different speakers. A total of 63 labels consisting of 61 phonemes plus a padding and blank symbol were used during both training and decoding. For evaluating phoneme error rates (PERs), the phoneme labels were mapped to 39 labels (Lee and Hon, 1989). The speech sequences were extracted with a 10ms hop size, a 25ms window size and were encoded with 40-dimensional Fourier-transform-based filterbanks plus energy. Their temporal first and second order differences were added, thus 123-dimensional vectors were used as inputs. All the input features were normalized to zero mean and unit variance per-speaker. The data splitting and feature extraction process was performed using Kaldi[1] (Povey et al., 2011) which is a C++ based speech recognition toolkit.

4.1.2. Implementation details

All evaluations were performed with the same settings when it comes to training CapsNets unless otherwise noted. Variables are initialized using a fan-avg method (Glorot and Bengio, 2010) from the uniform distribution, i.e. initially learning weights are drawn from $[-\sqrt{3 \times \alpha_s / n_{init}}, \sqrt{3 \times \alpha_s / n_{init}}]$, where n_{init} is the average of input and output unit numbers and the scaling factor α_s is set to 1.0. The first convolutional sub-block in a capsulation block consists of two 3×3 convolutional layers with 64 channels and a stride for both time and frequency dimensions is set to 2. The third sub-block consists of a 3×3 convolutional layer with stride one and their channels are set to P_D. Dropout (Srivastava et al., 2014) layers are applied after every layer at rate 0.2. Λ is set to one and we used 8 dimensional instantiation vectors. We utilized two kinds of normalization layers. First, batch normalization (Ioffe and Szegedy, 2015) layers are added after every convolutional layer in the first convolutional sub-block of a capsulation block. Second, layer normalization (Ba et al., 2016)

[1] `https://github.com/kaldi-asr/kaldi`

layers are also applied between every capsule layer. One modification is that layer normalization is performed not per each capsule but over all capsules in the same capsule group slice. An Adam optimizer (Kingma and Ba, 2015) is used for the gradient descent algorithm. A learning rate is updated for each step n_s depending on the two hyper-parameters, a warming-up step n_w and a scaling factor κ as follows:

$$\text{Learning Rate} = \kappa \times \text{rsqrt}(d_{emb}) \times min(n_s^{-0.5}, n_s \times n_w^{-1.5}) \tag{8}$$

, where rsqrt is the reciprocal of a square root. d_{emb} is a embedding size of Transformers and it is set to one for the CapsNets and n_w is set to 1,200. We applied an additional decay policy where a learning rate was started by setting κ to 0.5, and then κ was reduced to 0.1 after 27 epochs. We continued the training by 200 epochs to ensure sufficient weight updates. In order to avoid the accuracy being dependent on the early stop time, we evaluated PER with a model which is the averaged checkpoint of the last 10 epochs. Approximately 5,340-frames are contained in a batch for each training step according to their sequence length thus the learnable weight are updated 41,800 times per experiment. Lastly, beam width for decoding is set to 100.

The proposed system was implemented using Tensorflow (Abadi et al., 2016). The experiments were performed on a server equipped with an Intel 10900X 3.7GHz processor, 64GB main memory, and three graphic cards (one Titan RTX and two RTX 2080 Ti). We used a speech recognition scoring toolkit (SCTK)[2] built by the National Institute of Standards and Technology (NIST).

4.2. Performance comparisons

We first investigated the performance gain from the proposed routing algorithm using small CapsNet models with $L = 1$ and $P_H = 20$. Afterwards, we performed experiments using bigger models to find the competitive architectural configurations to compare with existing CTC-networks.

[2] `https://github.com/usnistgov/SCTK`

Routing Method	Iteration	PER(%)	
		Valid	Test
DR	1	26.1	27.1
	2	25.9	27.0
	3	26.0	27.1
SDR	1	25.0	26.2
	2	23.6	25.3
	3	24.5	26.0

Table 1: Performances depending on the routing methods and their iteration numbers

We compared the sequential version of DR (SDR) with its unmodified counterpart (DR) by performing experiments with 6 kinds of configurations as shown in Table 1. ω is set to one and the number of parameter of each model is 207,682. For both cases of DR and SDR, the iteration 2 version of models show the lowest PERs within each routing mechanism. The cases of iteration one and three show similar results for the test set (Test) in both cases. Among the DR models, the relative PER reduction (RPERR) is only 0.77% and 0.37% depending on the number of iterations for the validation set (Valid) and Test respectively. Every case of the SDR-based models showed better performances compared to their original versions. Even the worst SDR model which the iteration number is set to one shows 0.9% and 1.2% lower PERs for Valid and Test respectively compared to that of the best DR model. The RPERR among the SDR cases is at most 5.6% which is around 7 times larger than that of the DR models. We will discuss more about these experiments in Section 5 in order to explain how the routing mechanisms are operating by investigating the heat maps of c.

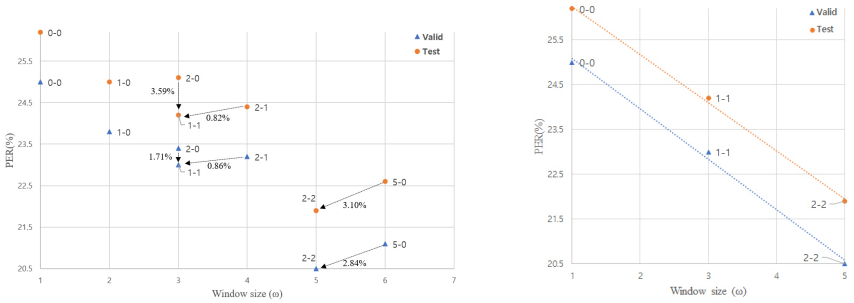
In order to investigate the changes in accuracy depending on the configurations related to ω, we compared 7 models as shown in Table 2. With the consideration of algorithmic delay caused by the right context size of windows ω_R, all models have longer or equal left context size ω_L to their ω_R. As ω is expanded from 1 to 6, the numbers of required parameters (Params.) are in-

Window		Params.	PER(%)	
Left	Right	(M)	Valid	Test
0	0	0.21	25.0	26.2
1	0	0.30	23.8	25.0
1	1	0.39	23.0	24.2
2	0	0.39	23.4	25.1
2	1	0.48	23.2	24.4
2	2	0.57	20.5	21.9
5	0	0.66	21.1	22.6

Table 2: Phoneme error rates (PERs) according to the left (ω_L) and right (ω_R) context size of window slices

creased from 0.21 to 0.66 million (M) as explained in Section 3.2. The RPERRs are at most 18.0% and 16.4% for Valid and Test respectively depending on ω settings as shown in the table. When both context sizes are set to two, the model shows the best performance with 0.9M less parameters compared to the case where the left context size is set to 5. The relationship between ω (X-axis) and PERs (Y-axis) are more explicitly described in Fig. 4. "2-1" indicates that ω_L and ω_R are 2 and 1 respectively and the exact figures of the corresponding PERs are in Table 2. Triangles and circles indicate Valid and Test respectively. We can see multiple cases where the balance of ω_L and ω_R has more of an effect on lowering the PERs than the number of parameters when $\omega > 1$ in Fig. 4a. This phenomenon is described by the dashed arrows in the figure and the percentages beside the arrows are RPERR between the balanced and unbalanced window configurations. Moreover, PERs of the three evenly sliced cases, that are "0-0", "1-1", and "2-2", show an almost linear error reduction according to their ω for both evaluation sets from 25.6% to 21.2% on average as shown in Fig 4b.

We also performed experiments on the capsule group depth, by doubling it continually from 2 to 16, to examine its effect as in Table 3. In these experiments,



(a) PERs depending on window size ω

(b) PERs by expanding the size of the balanced window slices

Figure 4: Phoneme error rates (PERs) depending on the configuration of window contexts)

Depth	Params. (M)	PER(%) Valid	PER(%) Test
2	0.14	26.7	28.0
4	0.19	24.1	25.2
8	0.39	23.0	24.2
16	1.15	20.8	22.1

Table 3: Phoneme error rates (PERs) depending on the depth of capsule groups

ω_L and ω_R are fixed to 1. As capsule group depths increased, the required Params. are also increased nearly proportional to the depths from 0.14M to 1.15M as explained in Section 3.2. The RPERRs for Valid and Test are 22.10% and 21.07% respectively. The PER reductions slow down at depth 4 from 2.6% and 2.8% to 1.1% and 1.0% for Valid and Test respectively, even though the parameter increase is 4 times larger from 0.5 to 2.

We compared the SRF models with the other CTC networks and RNN-T networks by stacking up the capsule layers which are made of the configurations, $P_H = 60$, $C_H = 30$ and $\omega = 3$ ($\omega_L = 1, \omega_R = 1$), as shown in Table 4. The BLSTM-based CTC network, which consists of 5 bi-directional layers of 250 LSTM cells (BLSTM-5L-250H), shows 1.2% better PRR for Test than its

Model	Online Decoding	Params. (M)	PER(%) Valid	PER(%) Test
BLSTM-5L-250H (Graves et al., 2013)	X	6.80	-	18.4
ULSTM-3L-421H (Graves et al., 2013)	O	3.80	-	19.6
CNN-10L-maxout (Zhang et al., 2016)	O	4.30	16.7	18.2
RNN-T-3L-250H (Graves et al., 2013)	X	4.30	-	17.7
TF-5L	X	1.99	19.4	20.0
TF-10L	X	3.63	17.6	18.8
TF-20L	X	6.93	17.1	18.4
SRF-1L	O	1.01	20.2	21.5
SRF-2L	O	0.99	19.2	21.3
SRF-5L	O	1.58	16.7	18.4
SRF-7L	O	1.97	16.0	17.4

Table 4: Phoneme error rates (PERs) comparisons depending on model structures

stream enabled version which is made of three uni-directional layers of 421 LSTM cells (ULSTM-3L-421H) (Graves et al., 2013). CNN-10L-maxout (Zhang et al., 2016), consists of 10 convolutional layers using maxout plus 1 max pooling layer after the first convolutional layer and is followed by three fully-connected layers activated by maxout. The network shows the best accuracy in PRR among the compared CTC networks at 83.3% and 81.8% on Valid and Test respectively. The RNN-T network (RNN-T-3L-250H) (Graves et al., 2013) consists of a pre-trained CTC network, which is composed of 3 bi-directional layers of 250 LSTM cells, and an additional hidden layer of 250 LSTM cells as a prediction network. RNN-T-3L-250H shows the best PRR of 82.3% with 4.3M parameters. We also performed the evaluations using Speech-Transformer (Dong et al., 2018)-based CTC networks which we implemented. Convolutional layers in front of Transformer encoders have the same structures with the first sub-block of the capsulation block as explained in Section 4.1.2. In each layer, d_{emb} is set to 128 and the dimension of position-wise fully connected feed-forward

layers is set to 1024. When it comes to learning rate decay, κ is set to 1.5 and reduced to 0.5 under the same condition as the cases of the CapsNets and n_w is set to 1,000. The number of attention headers is set to 4 and there are approximately 15,250 frames in a batch according to the length of utterances. In order to prevent overfitting, we set dropout rates for the input data, attention header, inner layers and residual connection to 0.3, 0.3, 0.4, and 0.4 respectively. The other configurations are the same as explained in Section 4.1.2. We added bigger penalties to non-diagonal elements in attention maps before applying the softmax function Eq. 1, depending on the distance δ from the diagonal of the maps in the form of $-\log(1 + \delta \times \beta)$ as described in (Dong et al., 2018), where β is a scaling factor and we set this to 1.0. TF-5L indicates that the Transformer consists of 5 encoder layers. The model consists of a similar number of parameters as the biggest SRF model at the bottom of the table. Although the model shows the lowest accuracy among the comparison models, it is easily improved by stacking the encoder layers. We compared 2 more Transformer-based CTC networks on Test with the LSTMs-based networks composed of a similar number of parameters. The Transformer model consisting of 10 layers shows 4.08% of RPERR with 0.17M less parameters compared to the ULSTM-based CTC network which has online decoding capability. TF-20L shows the same PER compared to the corresponding BLSTM-based CTC network with slightly (0.13M) more parameters.

The reason why SRF-1L requires 0.02M more parameters than SRF-2L is that SRF-1L directly connects primary capsule groups to class capsule groups thus SRF-1L needs 11,340 ($60 \times 63 \times 3$) transform matrices while SRF-2L needs 11,040 ($(60 \times 30 + 30 \times 63) \times 3$) transform matrices. When comparing the two models, the number of capsule layers seems to have more of an effect on improving the accuracy than the number of parameters as in Table 4. With 5 layers (SRF-5L), the model shows similar accuracy to the BLSTM-based CTC network while requiring 76.8% less parameters. By adding more layers, the proposed method finally attains 17.4% of PER with the model consisting of 7 capsule layers (SRF-7L) for Test. It is 2.2% more accurate than ULSTM-

3L-421H, which is the online capable version of LSTM, and 0.8% better PRR compared to CNN-based structure. Even more, the PER of SRF is slightly lower (RPERR is 1.65%) than that of RNN-T-3L-250H while keeping online decoding capability. Lastly, SRF requires 45.8% of parameters compared to both the CNN-based CTC network and the RNN-T.

5. Analysis

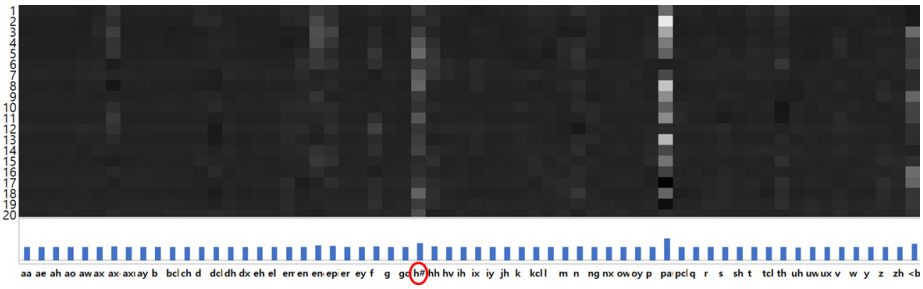


Figure 5: A heat map of coupling coefficients c (SDR, Iteration=1, t=137)

In this section, we investigate how the routing method works in SRF by observing the heat maps of c for a 5.47 seconds length utterance in the test set. In the heat maps, brighter cells refer to larger values ranging from 0.01 to 0.05. All models explained in this section are the same as explained in Table 1 of Section 4.2. Fig. 5 is a heat map of c which maps from primary capsules (vertical axis) to class capsules (horizontal axis) of the iteration 1 version of the SDR model at $t = 137$. A corresponding reference label is the red circled symbol "h#", which indicates the end of a sentence. The numbers on the vertical axis indicates primary capsule indexes. The bar graphs at the bottom of the figure represents the summation of coefficients per each class capsule. The summation of each row, i.e. the summation of coefficients per each primary capsule, is one and the coefficients which route capsules to the padding symbol are masked to zero as explained in Section 3.2 and they are not represented in the heat map. As shown in the figure, primary capsules are mostly routed to a class capsule corresponding to a "pau" symbol with the accumulated coefficient of 0.52 then

followed by “h#” with that of 0.41 and “<blank>” with that of 0.39. 7 primary capsules indexed by in the order of 17, 19, 20, 6, 7, 12 and 5 are routed more to the class capsule corresponding to the correct symbol “h#” rather than the capsule for “pau”.

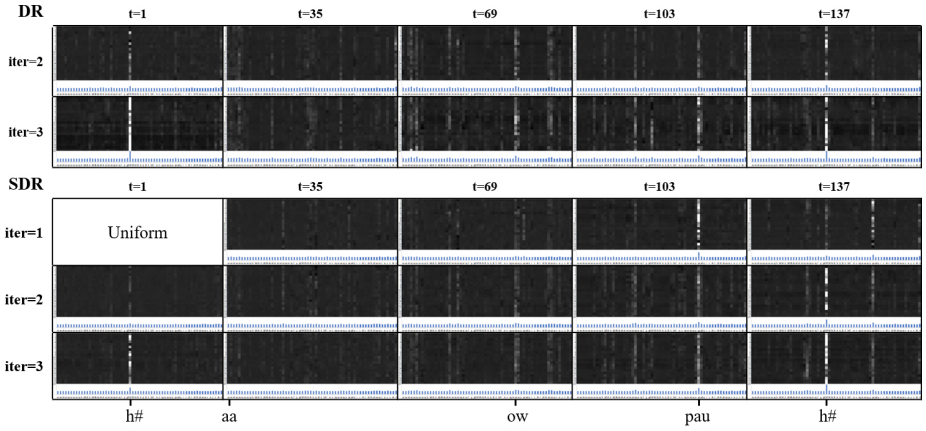
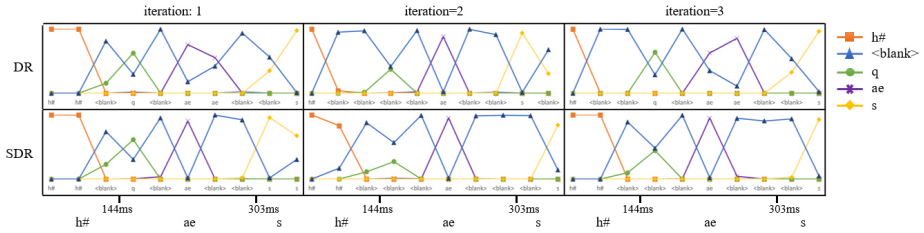
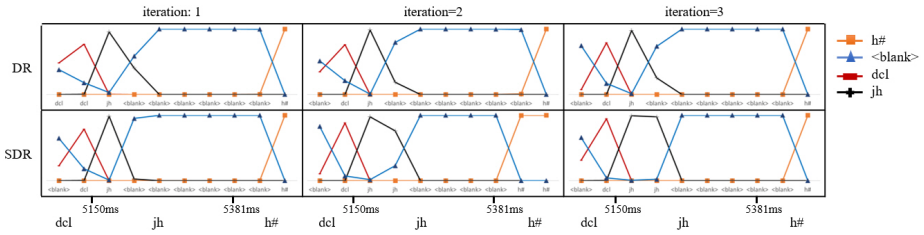


Figure 6: Heat maps of coupling coefficients c for DR and SDR

We compare 14 heat maps for different iteration numbers from 1 to 3 between DR and SDR as in Fig. 6. The coefficients of iteration 1 version of DR are all the same so it is not included in the figure. The reference symbols for each t are written in the bottom of the figure with the time markers. The majority of primary capsules are routed to the class capsule corresponding to the correct symbol. Among the two DR cases, the heat maps of the iteration 3 versions have a slightly higher contrast than that of the iteration 2 versions as shown in Fig. 6. In order words, as the number of iterations increases, it seems that the distributions become less uniform. The iteration 2 and 3 version of SDR models display the same phenomenon as the DR version. However, the SDR model with iteration 1 seems to have different behaviors besides that c is uniform at $t = 1$. The model routes the capsules to “pau” more than any other version at $t = 103$. Moreover, at $t = 137$, i.e. the end of the sentence, the model seems to route the majority of capsules to the class capsules corresponding to “pau” rather than the correct symbol “h#” as explained earlier in this section.



(a) t: $1 \rightarrow 10$



(b) t: $127 \rightarrow 137$

Figure 7: Softmax vectors depending on routing methods (X-axis: time (ms), Y-axis: probability)

In order to see how the phenomenon affects alignment, we checked the 10 probability vectors each at the beginning and end of the sentence as shown in Fig 7. Symbols are represented with different line shapes described in the right side of the figure. The labels on the X-axis of each graph indicate a symbol with the highest probability, i.e. the phoneme sequence of the greedy decoding. Reference time markers and symbols are written at the bottom of Fig 7a and Fig 7b. For all the cases, the misalignment phenomenon of the iteration 1 version of SDR model is not observed. It is not a big difference, but rather, the SDR model recognizes the symbol "s" as fast as the iteration 2 version DR as shown in Fig 7a. The end of sentence symbol "h#" is also precisely recognized in all the cases as in Fig 7b. Even in the case of SDR when iteration is set to 1 at $t = 137$, unlike the heat map where most of the primary capsules are routed to the class capsule to the symbol "pau", the probability of "h#" is the largest.

6. Related work

In this paper, we explored the potential capabilities of CapsNets for sequence encoding. There have been many research related to CapsNets such as improving their routing mechanisms or applying them to various kinds of tasks. In this section, we explain those attempts. The concept of routing between capsules was first introduced to recognize pose information (Hinton et al., 2011) and it was implemented in an auto encoder manner (Hinton and Zemel, 1993). DR (Sabour et al., 2017) is the earliest attempt to apply capsules for image classification problems. It showed better accuracy, not only in the original MNIST (LeCun and Cortes, 2010) dataset compared to CNNs, but also in highly overlapping digit cases in the MultiMNIST dataset. The accuracy on the overlapping cases was on par with that of sequential attention models. EM routing (Hinton et al., 2018) is a follow-up study of DR. It not only releases the length constraint of the instantiation vectors by defining activations as separate scalars but also reduces the size of transformation matrices to the square root size by representing the instantiation parameters as matrices. In addition, an expectation-maximization (EM) algorithm based on the Gaussian mixture model is applied to the routing method. Despite its structural and theoretical advances, the EM routing method has shown noncompetitive accuracy and computational complexity compared to DR (Malmgren, 2019; Hahn et al., 2019). This is the reason why we did not adopt it in this study, but it still has a suitable structure to be applied to the proposed method.

In order to improve implementations of the pioneer studies, various modifications were proposed. When it comes to iterative routing methods, since increasing the number of iterations can lead to unbalanced activations, (Wang and Liu, 2018) proposed an optimized DR method which applies an entropy regularizer to constrain the routing coefficient to be close to the uniform distribution. A min-max normalization (Zhao et al., 2019) was applied to resolve the performance degradation in DR caused by iterative usage of a softmax function as a normalization function for routing coefficients. For faster training, routing coef-

ficients were initialized from the learnable weights (Ramasinghe et al., 2018) and an attempt to introduce the EM algorithm to DR (Zhang et al., 2019b) under certain conditions was studied. Recently, CapsNet has been applied to relatively large datasets such as CiFAR 100 (Krizhevsky, 2009) by parallelizing iterative routing methods (Tsai et al., 2020) and showed competitive performance compared to ResNet (He et al., 2016). There is also self-routing (Hahn et al., 2019), which is a non-iterative routing method, that introduces the mixture-of-expect mechanism (Jacobs et al., 1991) to the routing method.

In addition to research which improves CapsNets themselves, there are various attempts to merge CapsNets or the routing mechanism with other models. Especially for sequence inputs, CapsNets are combined with existing sequential models either as a successor block at the top of the LSTM layers (Zhang et al., 2018; He et al., 2019a), the Transformer (Vaswani et al., 2017) encoder blocks (Liu et al., 2019), and bidirectional encoder representations from Transformers (BERT) (Devlin et al., 2019; Sun et al., 2020) or as an intermediate block in between encoders and decoders which are made of LSTMs (Wang, 2019). There are also attempts to put routing algorithms into attention methods and vice versa. Routing mechanisms are adopted into self-attention based models to cluster similar information from the multi-head attentions (Gu and Feng, 2019). In contrast, STAR-Caps (Ahmed and Torresani, 2019) merges attention methods into the routing mechanism.

CapsNets have been actively applied to a variety of fields because of their outstanding image encoding abilities. They are suitable for visual tracking (Ma and Wu, 2019) and object segmentation of medical images (LaLonde and Bagci, 2018). CapsNets also can be easily applied to non-visual tasks such as knowledge graph embedding and link prediction because of their representations of conceptual hierarchy relationships (Nguyen et al., 2019; Xinyi and Chen, 2019). For linguistic data, CapsNets were applied to text classification with k-means routing (Ren and Lu, 2018), machine translation in an encoder-decoder manner (Wang, 2019), user intent detection (Xia et al., 2018; Zhang et al., 2019a) and emotion detection using micro blogs (Zhong et al., 2020). Classification tasks

using audio and speech data (Jain, 2019) have also been actively researched to detect sound events (Iqbal et al., 2018; Vesperini et al., 2019) and classify emotions (Wu et al., 2019). Electrocardiogram signal categorization (Jayasekara et al., 2019) is another interesting task where CapsNets can be applied to classify input sequential data. Last but not least, isolated word recognition has been researched (Bae and Kim, 2018; Yan, 2018).

7. Conclusion

In this study, we proposed SRF, which is a novel framework to adapt CapsNets for encoding sequence data. We believe, this is the first capsule-only structure for seq2seq recognition. In the framework, input sequences are capsulized and sliced by the given window size. Routing from lower to higher levels is performed for each window slice by temporally sharing two kinds of parameters over a whole sequence. From the perspective of gradient-based optimization, the amount of required memory size can be controlled regardless of the length of input sequences by sharing the transformation matrix. Moreover, by initializing routing coefficients based on the routing output of the previous window slices, we could minimize decoding speed degradation caused by the routing iteration since the routing mechanism can be operated in a non-iterative manner during inference. The proposed method achieved competitive performance on a phoneme recognition task compared to CNNs, LSTMs, and Transformers in three aspects that are accuracy, amount of required memory, and streaming capabilities. An area of improvement for the future is the possibility of a fully end-to-end ASR system, which can represent linguistic context, based on CapsNet only architectures.

References

Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow IJ, Harp A, Irving G, Isard M, Jia Y, Józefowicz R, Kaiser L, Kudlur M, Levenberg J,

Mané D, Monga R, Moore S, Murray DG, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker PA, Vanhoucke V, Vasudevan V, Viégas FB, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. CoRR 2016;abs/1603.04467. URL: http://arxiv.org/abs/1603.04467. arXiv:1603.04467.

Ahmed K, Torresani L. Star-caps: Capsule networks with straight-through attentive routing. In: Wallach HM, Larochelle H, Beygelzimer A, d'Alché-Buc F, Fox EB, Garnett R, editors. Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada. 2019. p. 9098–107. URL: http://papers.nips.cc/paper/9110-star-caps-capsule-networks-with-straight-through-attentive-routing.

Ba LJ, Kiros JR, Hinton GE. Layer normalization. CoRR 2016;abs/1607.06450. URL: http://arxiv.org/abs/1607.06450. arXiv:1607.06450.

Bae J, Kim D. End-to-end speech command recognition with capsule network. In: Yegnanarayana B, editor. Interspeech 2018, 19th Annual Conference of the International Speech Communication Association, Hyderabad, India, 2-6 September 2018. ISCA; 2018. p. 776–80. URL: https://doi.org/10.21437/Interspeech.2018-1888. doi:10.21437/Interspeech.2018-1888.

Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. In: Bengio Y, LeCun Y, editors. 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. 2015. URL: http://arxiv.org/abs/1409.0473.

Chan W, Jaitly N, Le QV, Vinyals O. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing,

ICASSP 2016, Shanghai, China, March 20-25, 2016. IEEE; 2016. p. 4960–4. URL: https://doi.org/10.1109/ICASSP.2016.7472621. doi:10.1109/ICASSP.2016.7472621.

Devlin J, Chang M, Lee K, Toutanova K. BERT: pre-training of deep bidirectional transformers for language understanding. In: Burstein J, Doran C, Solorio T, editors. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers). Association for Computational Linguistics; 2019. p. 4171–86. URL: https://doi.org/10.18653/v1/n19-1423. doi:10.18653/v1/n19-1423.

Dong L, Xu S, Xu B. Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2018. p. 5884–8.

Garofolo JS, Lamel LF, Fisher WM, Fiscus JG, Pallett DS, Dahlgren NL. Darpa timit acoustic phonetic continuous speech corpus cdrom. 1993.

Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. In: Teh YW, Titterington DM, editors. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010. JMLR.org; volume 9 of *JMLR Proceedings*; 2010. p. 249–56. URL: http://proceedings.mlr.press/v9/glorot10a.html.

Goodfellow IJ, Warde-Farley D, Mirza M, Courville AC, Bengio Y. Maxout networks. CoRR 2013;abs/1302.4389. URL: http://arxiv.org/abs/1302.4389. arXiv:1302.4389.

Graves A, Fernández S, Gomez FJ, Schmidhuber J. Connectionist temporal ification: labelling unsegmented sequence data with recurrent neural

networks. In: Cohen WW, Moore AW, editors. Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006. ACM; volume 148 of *ACM International Conference Proceeding Series*; 2006. p. 369–76. URL: https://doi.org/10.1145/1143844.1143891. doi:10.1145/1143844.1143891.

Graves A, Mohamed A, Hinton GE. Speech recognition with deep recurrent neural networks. In: IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013. IEEE; 2013. p. 6645–9. URL: https://doi.org/10.1109/ICASSP.2013.6638947. doi:10.1109/ICASSP.2013.6638947.

Gu S, Feng Y. Improving multi-head attention with capsule networks. In: Tang J, Kan M, Zhao D, Li S, Zan H, editors. Natural Language Processing and Chinese Computing - 8th CCF International Conference, NLPCC 2019, Dunhuang, China, October 9-14, 2019, Proceedings, Part I. Springer; volume 11838 of *Lecture Notes in Computer Science*; 2019. p. 314–26. URL: https://doi.org/10.1007/978-3-030-32233-5_25. doi:10.1007/978-3-030-32233-5_25.

Hahn T, Pyeon M, Kim G. Self-routing capsule networks. In: Wallach HM, Larochelle H, Beygelzimer A, d'Alché-Buc F, Fox EB, Garnett R, editors. Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada. 2019. p. 7656–65. URL: http://papers.nips.cc/paper/8982-self-routing-capsule-networks.

He C, Peng L, Le Y, He J, Zhu X. Secaps: A sequence enhanced capsule model for charge prediction. In: Tetko IV, Kurková V, Karpov P, Theis FJ, editors. Artificial Neural Networks and Machine Learning - ICANN 2019: Text and Time Series - 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17-19, 2019, Proceedings, Part IV. Springer; volume 11730 of *Lecture Notes in Computer Science*; 2019a. p. 227–

39. URL: https://doi.org/10.1007/978-3-030-30490-4_19. doi:10.1007/978-3-030-30490-4_19.

He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. IEEE Computer Society; 2016. p. 770–8. URL: https://doi.org/10.1109/CVPR.2016.90. doi:10.1109/CVPR.2016.90.

He Y, Sainath TN, Prabhavalkar R, McGraw I, Alvarez R, Zhao D, Rybach D, Kannan A, Wu Y, Pang R, Liang Q, Bhatia D, Shangguan Y, Li B, Pundak G, Sim KC, Bagby T, Chang S, Rao K, Gruenstein A. Streaming end-to-end speech recognition for mobile devices. In: IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019. IEEE; 2019b. p. 6381–5. URL: https://doi.org/10.1109/ICASSP.2019.8682336. doi:10.1109/ICASSP.2019.8682336.

Hinton GE, Krizhevsky A, Wang SD. Transforming auto-encoders. In: Honkela T, Duch W, Girolami MA, Kaski S, editors. Artificial Neural Networks and Machine Learning - ICANN 2011 - 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I. Springer; volume 6791 of *Lecture Notes in Computer Science*; 2011. p. 44–51. URL: https://doi.org/10.1007/978-3-642-21735-7_6. doi:10.1007/978-3-642-21735-7_6.

Hinton GE, Sabour S, Frosst N. Matrix capsules with EM routing. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net; 2018. URL: https://openreview.net/forum?id=HJWLfGWRb.

Hinton GE, Zemel RS. Autoencoders, minimum description length and helmholtz free energy. In: Cowan JD, Tesauro G, Alspector J, editors. Advances in Neural Information Processing Systems 6, [7th NIPS Conference, Denver, Colorado, USA, 1993]. Morgan Kaufmann; 1993. p.

3–10. URL: http://papers.nips.cc/paper/798-autoencoders-minimum-description-length-and-helmholtz-free-energy.

Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computation 1997;9(8):1735–80. URL: https://doi.org/10.1162/neco.1997.9.8.1735. doi:10.1162/neco.1997.9.8.1735.

Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Bach FR, Blei DM, editors. Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015. JMLR.org; volume 37 of *JMLR Workshop and Conference Proceedings*; 2015. p. 448–56. URL: http://proceedings.mlr.press/v37/ioffe15.html.

Iqbal T, Xu Y, Kong Q, Wang W. Capsule routing for sound event detection. In: 26th European Signal Processing Conference, EUSIPCO 2018, Roma, Italy, September 3-7, 2018. IEEE; 2018. p. 2255–9. URL: https://doi.org/10.23919/EUSIPCO.2018.8553198. doi:10.23919/EUSIPCO.2018.8553198.

Jacobs RA, Jordan MI, Nowlan SJ, Hinton GE. Adaptive mixtures of local experts. Neural Computation 1991;3(1):79–87. URL: https://doi.org/10.1162/neco.1991.3.1.79. doi:10.1162/neco.1991.3.1.79.

Jain R. Improving performance and inference on audio classification tasks using capsule networks. CoRR 2019;abs/1902.05069. URL: http://arxiv.org/abs/1902.05069. arXiv:1902.05069.

Jayasekara H, Jayasundara V, Rajasegaran J, Jayasekara S, Seneviratne S, Rodrigo R. Timecaps: Capturing time series data with capsule networks. ArXiv 2019;abs/1911.11800.

Jia Y, Weiss RJ, Biadsy F, Macherey W, Johnson M, Chen Z, Wu Y. Direct speech-to-speech translation with a sequence-to-sequence model. In: Kubin G, Kacic Z, editors. Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19

September 2019. ISCA; 2019. p. 1123–7. URL: https://doi.org/10.21437/Interspeech.2019-1951. doi:10.21437/Interspeech.2019-1951.

Kim K, Jung S, Lee J, Han M, Kim C, Lee K, Gowda D, Park J, Kim S, Jin S, Lee Y, Yeo J, Kim D. Attention based on-device streaming speech recognition with large speech corpus. In: IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2019, Singapore, December 14-18, 2019. IEEE; 2019. p. 956–63. URL: https://doi.org/10.1109/ASRU46091.2019.9004027. doi:10.1109/ASRU46091.2019.9004027.

Kim M, Chi S. Detection of centerline crossing in abnormal driving using capsnet. J Supercomput 2019;75(1):189–96. URL: https://doi.org/10.1007/s11227-018-2459-6. doi:10.1007/s11227-018-2459-6.

Kingma DP, Ba J. Adam: A method for stochastic optimization. In: Bengio Y, LeCun Y, editors. 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. 2015. URL: http://arxiv.org/abs/1412.6980.

Krizhevsky A. Learning multiple layers of features from tiny images. Technical Report; 2009.

LaLonde R, Bagci U. Capsules for object segmentation. CoRR 2018;abs/1804.04241. URL: http://arxiv.org/abs/1804.04241. arXiv:1804.04241.

LeCun Y, Cortes C. MNIST handwritten digit database 2010;URL: http://yann.lecun.com/exdb/mnist/.

Lee K, Hon H. Speaker-independent phone recognition using hidden markov models. IEEE Trans Acoust Speech Signal Process 1989;37(11):1641–8. URL: https://doi.org/10.1109/29.46546. doi:10.1109/29.46546.

Lenssen JE, Fey M, Libuschewski P. Group equivariant capsule networks. In: Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett

R, editors. Advances in Neural Information Processing Systems 31. Curran Associates, Inc.; 2018. p. 8844–53. URL: http://papers.nips.cc/paper/8100-group-equivariant-capsule-networks.pdf.

Liu J, Lin H, Liu X, Xu B, Ren Y, Diao Y, Yang L. Transformer-based capsule network for stock movement prediction. In: Proceedings of the First Workshop on Financial Technology and Natural Language Processing. Macao, China; 2019. p. 66–73. URL: https://www.aclweb.org/anthology/W19-5511.

Ma D, Wu X. Tcdcaps: Visual tracking via cascaded dense capsules. CoRR 2019;abs/1902.10054. URL: http://arxiv.org/abs/1902.10054. arXiv:1902.10054; withdrawn.

Malmgren C. A Comparative Study of Routing Methods in Capsule Networks. Master's thesis; Linköping University, Computer Vision; 2019.

Moritz N, Hori T, Le J. Streaming automatic speech recognition with the transformer model. In: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2020. p. 6074–8.

Moritz N, Hori T, Roux JL. Streaming end-to-end speech recognition with joint ctc-attention based models. In: IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2019, Singapore, December 14-18, 2019. IEEE; 2019. p. 936–43. URL: https://doi.org/10.1109/ASRU46091.2019.9003920. doi:10.1109/ASRU46091.2019.9003920.

Nguyen DQ, Vu T, Nguyen TD, Nguyen DQ, Phung DQ. A capsule network-based embedding model for knowledge graph completion and search personalization. In: Burstein J, Doran C, Solorio T, editors. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers). Association for Computational Linguistics; 2019. p. 2180–9. URL: https://doi.org/10.18653/v1/n19-1226. doi:10.18653/v1/n19-1226.

Povey D, Ghoshal A, Boulianne G, Burget L, Glembek O, Goel NK, Hannemann M, Motlícek P, Qian Y, Schwarz P, Silovský J, Stemmer G, Veselý K. The kaldi speech recognition toolkit. 2011. .

Rabiner LR. A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE 1989;77(2):257–86.

Ramasinghe S, Athuraliya CD, Khan SH. A context-aware capsule network for multi-label classification. In: Leal-Taixé L, Roth S, editors. Computer Vision - ECCV 2018 Workshops - Munich, Germany, September 8-14, 2018, Proceedings, Part III. Springer; volume 11131 of *Lecture Notes in Computer Science*; 2018. p. 546–54. URL: https://doi.org/10.1007/978-3-030-11015-4_40. doi:10.1007/978-3-030-11015-4_40.

Ren H, Lu H. Compositional coding capsule network with k-means routing for text classification. CoRR 2018;abs/1810.09177. URL: http://arxiv.org/abs/1810.09177. arXiv:1810.09177.

Sabour S, Frosst N, Hinton GE. Dynamic routing between capsules. In: Guyon I, von Luxburg U, Bengio S, Wallach HM, Fergus R, Vishwanathan SVN, Garnett R, editors. Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA. 2017. p. 3856–66. URL: http://papers.nips.cc/paper/6975-dynamic-routing-between-capsules.

Srivastava N, Hinton GE, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 2014;15(1):1929–58. URL: http://dl.acm.org/citation.cfm?id=2670313.

Srivastava S, Agarwal P, Shroff G, Vig L. Hierarchical capsule based neural network architecture for sequence labeling. In: 2019 International Joint Conference on Neural Networks (IJCNN). 2019. p. 1–8.

Sun C, Yang Z, Wang L, Zhang Y, Lin H, Wang J. Attention guided capsule networks for chemical-protein interaction extraction. J Biomed Informat-

ics 2020;103:103392. URL: https://doi.org/10.1016/j.jbi.2020.103392. doi:10.1016/j.jbi.2020.103392.

Sutskever I, Vinyals O, Le QV. Sequence to sequence learning with neural networks. In: Ghahramani Z, Welling M, Cortes C, Lawrence ND, Weinberger KQ, editors. Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada. 2014. p. 3104–12. URL: http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.

Tsai YH, Srivastava N, Goh H, Salakhutdinov R. Capsules with inverted dot-product attention routing. In: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net; 2020. URL: https://openreview.net/forum?id=HJe6uANtwH.

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I. Attention is all you need. In: Guyon I, von Luxburg U, Bengio S, Wallach HM, Fergus R, Vishwanathan SVN, Garnett R, editors. Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA. 2017. p. 5998–6008. URL: http://papers.nips.cc/paper/7181-attention-is-all-you-need.

Vesperini F, Gabrielli L, Principi E, Squartini S. Polyphonic sound event detection by using capsule neural networks. J Sel Topics Signal Processing 2019;13(2):310–22. URL: https://doi.org/10.1109/JSTSP.2019.2902305. doi:10.1109/JSTSP.2019.2902305.

Wang D, Liu Q. An optimization view on dynamic routing between capsules. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings. OpenReview.net; 2018. URL: https://openreview.net/forum?id=HJjtFYJDf.

Wang M. Towards linear time neural machine translation with capsule networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Hong Kong, China: Association for Computational Linguistics; 2019. p. 803–12. URL: https://www.aclweb.org/anthology/D19-1074. doi:10.18653/v1/D19-1074.

Wu X, Liu S, Cao Y, Li X, Yu J, Dai D, Ma X, Hu S, Wu Z, Liu X, Meng H. Speech emotion recognition using capsule networks. In: IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019. IEEE; 2019. p. 6695–9. URL: https://doi.org/10.1109/ICASSP.2019.8683163. doi:10.1109/ICASSP.2019.8683163.

Xia C, Zhang C, Yan X, Chang Y, Yu PS. Zero-shot user intent detection via capsule neural networks. In: Riloff E, Chiang D, Hockenmaier J, Tsujii J, editors. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018. Association for Computational Linguistics; 2018. p. 3090–9. URL: https://doi.org/10.18653/v1/d18-1348. doi:10.18653/v1/d18-1348.

Xinyi Z, Chen L. Capsule graph neural network. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net; 2019. URL: https://openreview.net/forum?id=Byl8BnRcYm.

Yan X. Using Capsule Networks for Image and Speech Recognition Problems. Master's thesis; Arizona State University, Electrical engineering; 2018.

Yin J, Jiang X, Lu Z, Shang L, Li H, Li X. Neural generative question answering. In: Kambhampati S, editor. Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016. IJCAI/AAAI Press; 2016. p. 2972–8. URL: http://www.ijcai.org/Abstract/16/422.

Zeghidour N, Xu Q, Liptchinsky V, Usunier N, Synnaeve G, Collobert R. Fully convolutional speech recognition. CoRR 2018;abs/1812.06864. URL: http://arxiv.org/abs/1812.06864. arXiv:1812.06864.

Zhang C, Li Y, Du N, Fan W, Yu PS. Joint slot filling and intent detection via capsule neural networks. In: Korhonen A, Traum DR, Màrquez L, editors. Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers. Association for Computational Linguistics; 2019a. p. 5259–67. URL: https://doi.org/10.18653/v1/p19-1519. doi:10.18653/v1/p19-1519.

Zhang N, Deng S, Sun Z, Chen X, Zhang W, Chen H. Attention-based capsule networks with dynamic routing for relation extraction. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. Brussels, Belgium: Association for Computational Linguistics; 2018. p. 986–92. URL: https://www.aclweb.org/anthology/D18-1120. doi:10.18653/v1/D18-1120.

Zhang S, Zhou Q, Wu X. Fast dynamic routing based on weighted kernel density estimation. In: Lu H, editor. Cognitive Internet of Things: Frameworks, Tools and Applications. Springer; volume 810 of *Studies in Computational Intelligence*; 2019b. p. 301–9. URL: https://doi.org/10.1007/978-3-030-04946-1_30. doi:10.1007/978-3-030-04946-1_30.

Zhang Y, Pezeshki M, Brakel P, Zhang S, Laurent C, Bengio Y, Courville AC. Towards end-to-end speech recognition with deep convolutional neural networks. In: Morgan N, editor. Interspeech 2016, 17th Annual Conference of the International Speech Communication Association, San Francisco, CA, USA, September 8-12, 2016. ISCA; 2016. p. 410–4. URL: https://doi.org/10.21437/Interspeech.2016-1446. doi:10.21437/Interspeech.2016-1446.

Zhao Z, Kleinhans A, Sandhu G, Patel I, Unnikrishnan KP. Capsule networks with max-min normalization. CoRR 2019;abs/1903.09662. URL: http://arxiv.org/abs/1903.09662. arXiv:1903.09662.

Zhong X, Liu J, Li L, Chen S, Lu W, Dong Y, Wu B, Zhong L. An emotion classification algorithm based on spt-capsnet. Neural Computing and Applications 2020;32(7):1823–37. URL: https://doi.org/10.1007/s00521-019-04621-y. doi:10.1007/s00521-019-04621-y.